Speak Less, Hear Enough: On Dynamic Announcement Intervals in Wireless On-demand Networks

L. Baumgärtner, P. Graubner, J. Höchst, A. Klein, B. Freisleben

WONS2017 — February 21 - 24, 2017

Table of Contents

Announcement protocols

Network protocols relying on broadcasting announcements:

Announcement protocols

Network protocols relying on broadcasting announcements:

- Service Discovery: *Bonjour / ZeroConf*
- Routing Protocols: *RIP, OLSR*
- Delay-tolerant Networking (DTN): *Forban, Serval*

Announcement protocols

Network protocols relying on broadcasting announcements:

- Service Discovery: *Bonjour / ZeroConf*
- Routing Protocols: *RIP, OLSR*
- Delay-tolerant Networking (DTN): *Forban, Serval*

Bandwidth in wireless networks (802.11, Bluetooth) is limited.

DTN in emergency communication

DTN in emergency communication

- **Fast spreading** of messages and files produced at a disaster site.

DTN in emergency communication

- **Fast spreading** of messages and files produced at a disaster site.
- **Epidemic routing** to as many neighbors as possible.

DTN in emergency communication

- **Fast spreading** of messages and files produced at a disaster site.
- **Epidemic routing** to as many neighbors as possible.
- Static nodes (*islands*):
  - people trapped in houses, emergency camps, etc.

DTN in emergency communication

- **Fast spreading** of messages and files produced at a disaster site.
- **Epidemic routing** to as many neighbors as possible.
- Static nodes (*islands*):
  - people trapped in houses, emergency camps, etc.
- Moving nodes (*carrier-pigeons*):
  - by bike, car, foot, etc.

Delay-tolerant data exchange

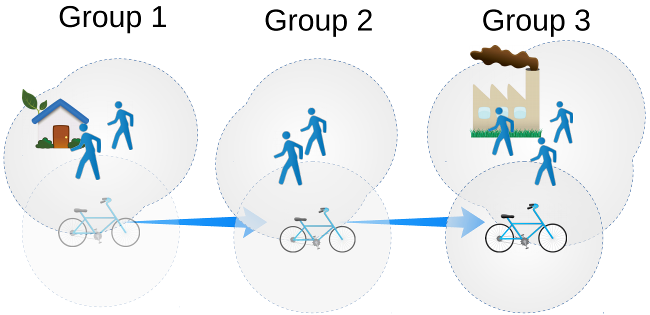


Figure: Drive-by store-and-forward data exchange.

Delay-tolerant data exchange

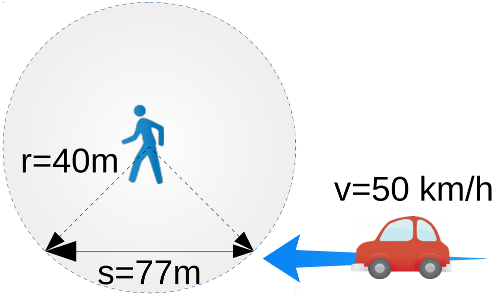


Figure: Drive-by window of opportunity example.

DTN drawbacks

- $r = 40m$: WiFi radius
- $d = 10m$: Node-to-street distance
- $v = 50km/h$: drive-by speed

$\rightarrow$ under 6 seconds for node discovery and exchange of data.

High announcement rates: **more** power consumed,
low announcement rates: data exchange **time reduced**.

Basic idea

Regular static announcements:
Announce myself to the other nodes within a **fixed time delay**.

Dynamic announcements:
Adapt announcement rate **dynamically**, based on multiple properties.

Interface for announcement computation

Access to a few general purpose variables:

Interface for announcement computation

Access to a few general purpose variables:

- current announcement delay

Interface for announcement computation

Access to a few general purpose variables:

- current announcement delay
- global announcement count

Interface for announcement computation

Access to a few general purpose variables:

- current announcement delay
- global announcement count
- current number of unique peers

Announcement interval computation strategies

1. *Static*: **fixed** 2s announcement interval

Announcement interval computation strategies

1. *Static*: **fixed** 2s announcement interval
2. *Random*: **random** announcement interval

Announcement interval computation strategies

1. *Static*: **fixed** 2s announcement interval
2. *Random*: **random** announcement interval
3. *RandomSweet*: new random interval, if **global count is *bad***

Announcement interval computation strategies

1. *Static*: **fixed** 2s announcement interval
2. *Random*: **random** announcement interval
3. *RandomSweet*: new random interval, if **global count is *bad***
4. *Step*: raise / lower node announce interval **step-by-step**

Announcement interval computation strategies

1. *Static*: **fixed** 2s announcement interval
2. *Random*: **random** announcement interval
3. *RandomSweet*: new random interval, if **global count is *bad***
4. *Step*: raise / lower node announce interval **step-by-step**
5. *StepRand*: *Step* with added **small random number**

Announcement interval computation strategies

1. *Static*: **fixed** 2s announcement interval
2. *Random*: **random** announcement interval
3. *RandomSweet*: new random interval, if **global count is *bad***
4. *Step*: raise / lower node announce interval **step-by-step**
5. *StepRand*: *Step* with added **small random number**
6. *MaxFirst*: **defensive** - set to low rate and raise step-by-step

Announcement interval computation strategies

1. *Static*: **fixed** 2s announcement interval
2. *Random*: **random** announcement interval
3. *RandomSweet*: new random interval, if **global count is *bad***
4. *Step*: raise / lower node announce interval **step-by-step**
5. *StepRand*: *Step* with added **small random number**
6. *MaxFirst*: **defensive** - set to low rate and raise step-by-step
7. *MinFirst*: **aggressive** - set to high rate and lower step-by-step

Announcement interval computation strategies

1. *Static*: **fixed** 2s announcement interval
2. *Random*: **random** announcement interval
3. *RandomSweet*: new random interval, if **global count is *bad***
4. *Step*: raise / lower node announce interval **step-by-step**
5. *StepRand*: *Step* with added **small random number**
6. *MaxFirst*: **defensive** - set to low rate and raise step-by-step
7. *MinFirst*: **aggressive** - set to high rate and lower step-by-step
8. *Unsteady*: delay derived directly from the **number of nodes**

Implementation constraints

Observation delay:

Implementation constraints

Observation delay:

- **compute announcement interval** afterwards

Implementation constraints

Observation delay:

- **compute announcement interval** afterwards
- announce **at least once** per observation delay

Implementation constraints

Observation delay:

- **compute announcement interval** afterwards
- announce **at least once** per observation delay
- **globally** defined for all nodes

Implementation constraints

Observation delay:

- **compute announcement interval** afterwards
- announce **at least once** per observation delay
- **globally** defined for all nodes
- the higher, the longer the network needs to adapt to new situations

Implementation constraints

Observation delay:

- **compute announcement interval** afterwards
- announce **at least once** per observation delay
- **globally** defined for all nodes
- the higher, the longer the network needs to adapt to new situations
- 20 seconds used in this paper

Quality measuring properties

Main goal: 1 second global announcement delay

Quality measuring properties

Main goal: 1 second global announcement delay

- *Global Anouncement Rate*: announcements per second

Quality measuring properties

Main goal: 1 second global announcement delay

- *Global Anouncement Rate*: announcements per second
- (*Global Announcement Gaps*: time between two announcements)

Quality measuring properties

Main goal: 1 second global announcement delay

- *Global Anouncement Rate*: announcements per second
- (*Global Announcement Gaps*: time between two announcements)
- *Adaptation Rate*: time needed to adapt to the new rate

Example Application: Mesher

Example Application: Mesher

- simple local chat, written in Google's Go

Example Application: Mesher

- simple local chat, written in Google's Go
- 642 bytes broadcast packets for neighbor discovery and database status updates

Example Application: Mesher

- simple local chat, written in Google's Go
- 642 bytes broadcast packets for neighbor discovery and database status updates
- *JavaScript*-based API for dynamic announcement computation

Evaluation setup: network emulation

Evaluation setup: network emulation

- ***Centralized*** *network*: all nodes connected centrally

Evaluation setup: network emulation

- ***Centralized*** *network*: all nodes connected centrally
- ***Growing*** *network*: nodes added periodically

Evaluation setup: network emulation

- ***Centralized*** *network*: all nodes connected centrally
- ***Growing*** *network*: nodes added periodically
- ***Merging*** *network*: merge of two equally sized networks

Evaluation setup: network emulation

- ***Centralized*** *network*: all nodes connected centrally
- ***Growing*** *network*: nodes added periodically
- ***Merging*** *network*: merge of two equally sized networks
- ***Splitting*** *network*: split into two equally sized networks

Evaluation setup: physical testbed

Evaluation setup: physical testbed

- **Raspberry Pi 3** Model B single-board computers

Evaluation setup: physical testbed

- **Raspberry Pi 3** Model B single-board computers
- Vendor-provided Debian-based **Raspbian OS**

Evaluation setup: physical testbed

- **Raspberry Pi 3** Model B single-board computers
- Vendor-provided Debian-based **Raspbian OS**
- **8 network participants**

Evaluation setup: physical testbed

- **Raspberry Pi 3** Model B single-board computers
- Vendor-provided Debian-based **Raspbian OS**
- **8 network participants**
- **1 system under test** (SUT)

Evaluation setup: physical testbed

- **Raspberry Pi 3** Model B single-board computers
- Vendor-provided Debian-based **Raspbian OS**
- **8 network participants**
- **1 system under test** (SUT)
- Data-logging at 5 Hz using an ***Odroid Smart Power***

Test configurations

Test configurations

- **Eight** announcement **strategies**

Test configurations

- **Eight** announcement **strategies**
- **Number of nodes**: 2, 5, 10, 25, 50, 100, 200

Test configurations

- **Eight** announcement **strategies**
- **Number of nodes**: 2, 5, 10, 25, 50, 100, 200
- **batch** node start, **delayed** node start

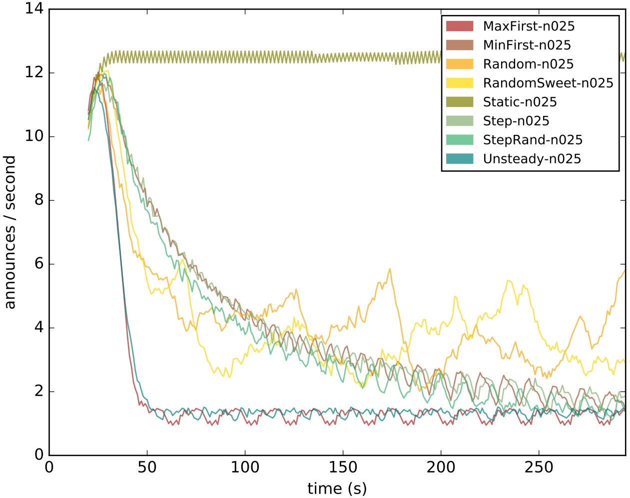
Test configurations

- **Eight** announcement **strategies**
- **Number of nodes**: 2, 5, 10, 25, 50, 100, 200
- **batch** node start, **delayed** node start
- **two dynamic** network configurations: *Split* and *Merge*

Test configurations

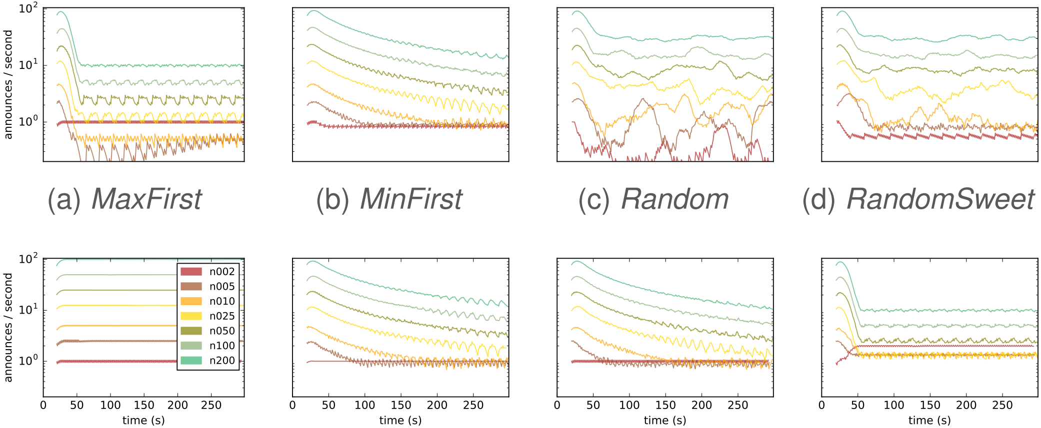
- **Eight** announcement **strategies**
- **Number of nodes**: 2, 5, 10, 25, 50, 100, 200
- **batch** node start, **delayed** node start
- **two dynamic** network configurations: *Split* and *Merge*
- total of **224 independent experiments**

Announcements in a 25 node static network (1)



Announcements in a 25 node static network (2)

All non-random strategies reach the goal of a **less saturated network** and also approach **the same minimum**.



(a) *MaxFirst* (b) *MinFirst* (c) *Random* (d) *RandomSweet*

(e) *Static* (f) *Step* (g) *StepRand* (h) *Unsteady*

Adaptation rate

Adaptation rate

- *Unsteady* and *MaxFirst* show **very high adaptation rates**, since the announcement delay is set after the first observation delay.

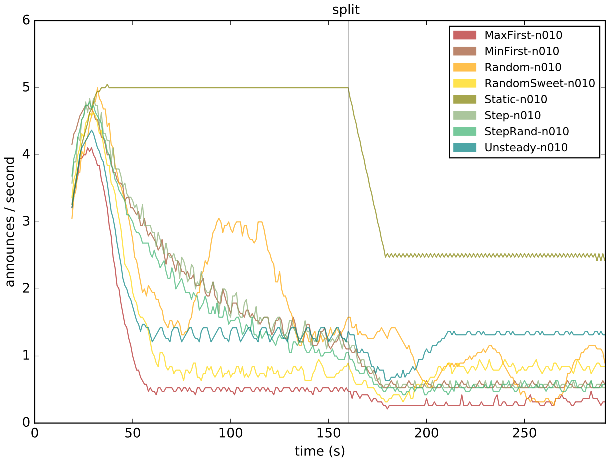
Adaptation rate

- *Unsteady* and *MaxFirst* show **very high adaptation rates**, since the announcement delay is set after the first observation delay.
- *MaxFirst* achieves a **high** rate in **larger** islands, while *MinFirst* achieves a **higher** adaptation rate in **smaller** islands.

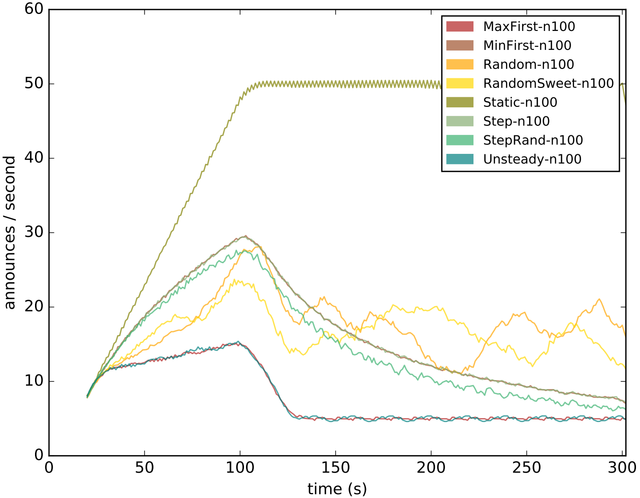
Adaptation rate

- *Unsteady* and *MaxFirst* show **very high adaptation rates**, since the announcement delay is set after the first observation delay.
- *MaxFirst* achieves a **high** rate in **larger** islands, while *MinFirst* achieves a **higher** adaptation rate in **smaller** islands.
- Adaptation rates of *Step*-based strategies **depend on the number of nodes**.

Adaptation rate: 10 nodes split



Adaptation rate: 100 nodes delayed



Name \ # Nodes	2	5	10	25	50
Static	291	732	1460	3658	7296
Random	34.4%	47.0%	37.0%	37.9%	37.3%
RandSweet	58.1%	41.7%	29.0%	35.6%	37.7%
Step	101.7%	**45.4%**	**35.2%**	**33.2%**	**33.4%**
StepRand	99.7%	42.5%	32.5%	30.1%	30.2%
MaxFirst	99.0%	21.2%	**17.1%**	**17.0%**	**17.1%**
MinFirst	84.9%	44.3%	34.7%	33.3%	33.5%
Unsteady	**188.7%**	56.8%	**32.5%**	**17.7%**	**17.1%**

Table: Bandwidth Comparison

Bandwidth savings

Bandwidth savings

- *Step*, *StepRand* and *MinFirst*: **bandwidth savings** $>$ **60%.**

Bandwidth savings

- *Step*, *StepRand* and *MinFirst*: **bandwidth savings** $>$ **60%.**
- *Unsteady* and *MaxFirst*: **bandwidth savings** $>$ **80%**
  $\rightarrow$ quick adaptation to the given situations.

Bandwidth savings

- *Step*, *StepRand* and *MinFirst*: **bandwidth savings** $>$ **60%.**
- *Unsteady* and *MaxFirst*: **bandwidth savings** $>$ **80%**
  $\rightarrow$ quick adaptation to the given situations.
- Unsteady: *188.7%* of *Static* in a two nodes network.
  $\rightarrow$ low announcement delays in small networks achievable.

Energy consumption: setup

Energy consumption: setup

- 8 regular nodes, **1 system under test**

Energy consumption: setup

- 8 regular nodes, **1 system under test**
- **ad-hoc** (1.35 W idle consumption) and **managed** mode (1.45 W)

Energy consumption: setup

- 8 regular nodes, **1 system under test**
- **ad-hoc** (1.35 W idle consumption) and **managed** mode (1.45 W)
- added Static05 and Static01, with 2 / 10 announces per second

Energy consumption: setup

- 8 regular nodes, **1 system under test**
- **ad-hoc** (1.35 W idle consumption) and **managed** mode (1.45 W)
- added Static05 and Static01, with 2 / 10 announces per second

$$E := \int_0^{300} P_{measured}(t)\,\mathrm{d}t - 300 * P_{idle}$$

Name	# Ann.	E (mWh)	rel. Ann.	rel. E	ratio
Static	**1323**	**1.99**	1.00	1.00	1.00
Static05	5404	11.97	4.08	6,00	1.47
Static01	**29342**	**32.52**	22.18	16.31	0.74
MaxFirst	**256**	**1.17**	0.19	0.59	3.04
MinFirst	473	1.26	0.36	0.63	3.04
Random	434	1.34	0.33	0.67	2.04
RandomSweet	**342**	**0.73**	0.26	0.37	1.42
Step	495	1.20	0.37	0.60	1.61
StepRand	460	1.12	0.35	0.56	1.61
Unsteady	**514**	**1.38**	0.39	0.69	1.78

Energy consumption: overall results

Energy consumption: overall results

- General trend proven: **less announcements** → **less power consumed**

Energy consumption: overall results

- General trend proven: **less announcements** $\rightarrow$ **less power consumed**
- correlation coefficient $r = 0.985$

Energy consumption: overall results

- General trend proven: **less announcements** $\rightarrow$ **less power consumed**
- correlation coefficient $r = 0.985$
- **Side-effects** due to programming language, OS, ...

Energy consumption: overall results

- General trend proven: **less announcements** $\rightarrow$ **less power consumed**
- correlation coefficient $r = 0.985$
- **Side-effects** due to programming language, OS, ...
- though relatively small, **announcements effect battery lifetimes**

Conclusion

Conclusion

- **Eight different announcement strategies** compared

Conclusion

- **Eight different announcement strategies** compared
- **reduction by 80%** compared to a static strategy, while reaching the goal of a fast island discovery.

Conclusion

- **Eight different announcement strategies** compared
- **reduction by 80%** compared to a static strategy, while reaching the goal of a fast island discovery.
- **Energy impact**: announcements effect battery lifetimes and are **worth to be reduced**.

Future Work

Future Work

- More realistic **WiFi emulation**, eg. Island center vs. edge nodes

Future Work

- More realistic **WiFi emulation**, eg. Island center vs. edge nodes
- Design algorithms based on **additional information**

Future Work

- More realistic **WiFi emulation**, eg. Island center vs. edge nodes
- Design algorithms based on **additional information**
- Evaluate on **real world applications** eg. *Serval*

Future Work

- More realistic **WiFi emulation**, eg. Island center vs. edge nodes
- Design algorithms based on **additional information**
- Evaluate on **real world applications** eg. *Serval*
- *Make software use dynamic announcements.*

The final Slide

Thanks for your Attention!

Are there any questions?